Adaptive time-stepping for incompressible flow. Part II: Navier-Stokes equations

Kay, David A.; Gresho, Philip M.; Griffiths, David; Silvester, David J.

Published in:
SIAM Journal on Scientific Computing

DOI:
10.1137/080728032

Publication date:
2010

SIAM J. SCI. COMPUT.
Vol. 32, No. 1, pp. 111–128

ADAPTIVE TIME-STEPPING FOR INCOMPRESSIBLE FLOW PART II: NAVIER–STOKES EQUATIONS*

DAVID A. KAY†, PHILIP M. GRESHO‡, DAVID F. GRIFFITHS§, AND DAVID J. SILVESTER¶

Abstract. We outline a new class of robust and efficient methods for solving the Navier–Stokes equations. We describe a general solution strategy that has two basic building blocks: an implicit time integrator using a stabilized trapezoid rule with an explicit Adams–Bashforth method for error control, and a robust Krylov subspace solver for the spatially discretized system. We present numerical experiments illustrating the potential of our approach.

Key words. time-stepping, adaptivity, Navier–Stokes, preconditioning, fast solvers

AMS subject classifications. 65M12, 65M15, 65M20

DOI. 10.1137/080728032

1. Background and context. Simulation of the motion of an incompressible fluid remains an important but very challenging problem. The resources required for accurate three-dimensional simulation of practical flows test even the most advanced of supercomputer hardware. The effectiveness of our stabilized TR–AB2 time-stepping algorithm that we explore here is demonstrated in the context of convection-diffusion problems in part I of this work [8]. Therein it is shown that stabilized TR–AB2 is particularly well suited to long time integration of advection-dominated problems and is a very effective algorithm when faced with general advection-diffusion problems with different time scales governing the system evolution. In this paper, our focus is on assessing the performance of the integrator in combination with a state-of-the-art iterative solver in the context of method-of-lines discretization of the Navier–Stokes equations.

For simplicity, the case of a two-dimensional flow domain Ω is considered here. Our solver methodology is exactly the same in the case of a three-dimensional flow model. Thus, the flow domain boundary Γ consists of two nonoverlapping segments $\Gamma_D \cup \Gamma_N$ associated with specified flow and natural outflow boundary conditions, respectively,

$$\frac{\partial \vec{u}}{\partial t} - \nu \nabla^2 \vec{u} + \vec{u} \cdot \nabla \vec{u} + \nabla p = 0 \qquad \text{in } \Omega \times [0, T], \tag{1.1}$$

$$-\nabla \cdot \vec{u} = 0 \qquad \text{in } \Omega \times [0, T], \tag{1.2}$$

*Received by the editors June 21, 2008; accepted for publication (in revised form) February 9, 2009; published electronically February 5, 2010. This collaboration was supported by EPSRC grant GR/R26092/1.

http://www.siam.org/journals/sisc/32-1/72803.html

†Oxford University Computing Laboratory, Oxford, OX1 3QD, UK (dkay@comlab.ox.ac.uk).

‡Livermore, CA (pgresho@comcast.net).

§Division of Mathematics, University of Dundee, Dundee, DD1 4HN, Scotland, UK (dfg@maths.dundee.ac.uk).

¶School of Mathematics, University of Manchester, Manchester, M13 9PL, UK (d.silvester@manchester.ac.uk).

$$\vec{u} = \vec{g} \quad \text{on } \Gamma_D \times [0,T], \tag{1.3}$$

$$\nu \nabla \vec{u} \cdot \vec{n} - p\,\vec{n} = \vec{0} \quad \text{on } \Gamma_N \times [0,T], \tag{1.4}$$

$$\vec{u}(\vec{x},0) = \vec{u}_0(\vec{x}) \text{ in } \Omega. \tag{1.5}$$

Our notation is completely standard: $\vec{u}$ is the fluid velocity, p is the scalar pressure, $\nu > 0$ is a specified viscosity parameter (in a nondimensional setting it is the inverse of the Reynolds number), and $T > 0$ is some final time. The initial velocity field $\vec{u}_0$ is typically assumed to satisfy the incompressibility constraint, that is, $\nabla \cdot \vec{u}_0 = 0$. Unless stated otherwise, it is implicitly assumed that Γ_N has nonzero measure, in which case the pressure p is uniquely specified by the outflow boundary condition.

Conventional approaches to solving the initial value problem (1.1)–(1.5) typically use semi-implicit time integration leading to a Poisson or Stokes-type problem at every time step, but with a stability restriction on the time step. In contrast, there is no time-step restriction in our case. The price that must be paid for this improved robustness is the need to solve a so-called *Oseen* problem at every time step. Fortunately, very efficient solvers for Oseen problems have become a reality in the last decade; see, for example, [6], [13], [3], [2]. Specifically, the preconditioning framework that has evolved offers the possibility of uniformly fast convergence independent of the problem parameters (namely, the mesh size, the time step, and the Reynolds number).

A common viewpoint (see, for example, Turek [18]) is that a coupled solver is mainly of use for steady flows, whereas projection-type schemes are preferred when modeling unsteady flows. We aim to challenge this assertion. Of course, projection-type schemes can be very effective—especially if implemented using multigrid and combined with a fixed time-stepping strategy. Their limitation is the fact that decoupling the velocity and pressure inevitably leads to smaller time steps when compared to a coupled solver strategy. The big attraction of an implicit discretization in time is that it enables the possibility of self-adaptive time-step control, with time steps automatically chosen to "follow the physics."

An outline of the paper is as follows. The temporal and spatial discretization of (1.1)–(1.5) is discussed in section 2. The linear algebra aspects are discussed in section 3, and the performance of our solver methodology is assessed in section 4. We have tested our solver on a range of flow problems. Results for two benchmark flow problems are presented here: first, a driven cavity flow that ultimately reaches a steady state; and second, a developing flow around a cylinder that reaches a periodic state of vortex shedding. We hope that, at the end, the reader will be convinced not only that incompressible flow problems can be solved more efficiently using an adaptive time integrator but also that studying the behavior of the computed time step can help to delineate different phases of the evolution of the flow.

2. Discretization aspects. Our "basic" time-stepping algorithm is the well-known, second-order accurate, trapezoid rule (TR). Let the interval $[0,T]$ be divided into N steps, $\{t_i\}_{i=1}^{N}$, and let $\vec{v}^j$ denote $\vec{v}(\vec{x},t_j)$. The semidiscretized problem is the following: Given $(\vec{u}^n, p^n)$ at time level t_n and boundary data $\vec{g}^{n+1}$ at time level t_{n+1},

compute $(\vec{u}^{n+1}, p^{n+1})$ via[1]

$$\frac{2}{k_{n+1}}\,\vec{u}^{n+1} - \nu\nabla^2\,\vec{u}^{n+1} + \vec{u}^{n+1}\cdot\nabla\,\vec{u}^{n+1} + \nabla p^{n+1} = \frac{2}{k_{n+1}}\,\vec{u}^{n} + \frac{\partial\,\vec{u}^{\,n}}{\partial t} \quad \text{in } \Omega, \tag{2.1}$$

$$-\nabla\cdot\,\vec{u}^{n+1} = 0 \qquad \text{in } \Omega, \tag{2.2}$$

$$\vec{u}^{n+1} = \vec{g}^{n+1} \text{ on } \Gamma_D, \tag{2.3}$$

$$\nu\nabla\,\vec{u}^{n+1}\cdot\,\vec{n} - p^{n+1}\,\vec{n} = \vec{0} \text{ on } \Gamma_N. \tag{2.4}$$

Here, $k_{n+1} := t_{n+1} - t_n$ is the current time step, and $\frac{\partial\,\vec{u}\,^n}{\partial t} := \nu\nabla^2\,\vec{u}^n - \vec{u}^n\cdot\nabla\,\vec{u}^n - \nabla p^n$ is shorthand for the acceleration at time t_n.

The limited stability of TR time-stepping for the incompressible Navier–Stokes equations is extensively discussed in the literature, for example, in the well-cited paper by Simo and Armero [15]. The basic algorithm has some attractive features, however. In particular, solving a simple ODE model of convection-diffusion,

$$\dot{y} = -\left(\frac{1}{\tau} + \mathbf{i}\,\omega\right) y, \qquad y(0) = 1, \tag{2.5}$$

where τ corresponds to a decay time constant and ω is a frequency parameter, it is easily shown (see [8, sect. 2]) that TR is unconditionally stable (A-stable) and nondissipative. This is important when modeling pure advection ($\tau = \infty$), or even advection-*dominated* problems ($\frac{1}{\tau} \ll \omega$). Dettmer and Perić [1] critically compare TR with alternative time-stepping algorithms in the context of fixed time-step algorithms for convection-diffusion equations and for Navier–Stokes equations. They present results showing that the lack of numerical damping within TR can be problematic if the time step is kept fixed (and is not small enough). Smith and Silvester [14] draw similar conclusions when comparing fixed time-step TR with the three-stage operator-splitting methods advocated by Turek [18]. Such problems are circumvented if an *adaptive* time-step strategy is employed and the TR method is *stabilized* as described later.

From (2.1) it is evident that a numerical scheme for handling the nonlinear term $\vec{u}^{n+1}\cdot\nabla\,\vec{u}^{n+1}$ is needed at every time step. A standard approach (see Gresho and Sani [9, p. 800]) would be solve the system (2.1)–(2.4) to a predefined accuracy using some variant of the Newton iteration. Although this requires inner iterations, the approach may still be cost-effective if it avoids any loss of stability which, using self-adaptive time-stepping, usually leads to a reduction in the time-step size. We advocate an alternative approach in this work—computational experiments in the final section show that if the linearization is done using $\vec{u}^{n+1}\cdot\nabla\,\vec{u}^{n+1} \approx \vec{w}^{n+1}\cdot\nabla\,\vec{u}^{n+1}$, where

$$\vec{w}^{n+1} = (1 + (k_{n+1}/k_n))\,\vec{u}^n - (k_{n+1}/k_n)\,\vec{u}^{n-1}, \tag{2.6}$$

then temporal stability is not compromised significantly. The linearization (2.6) is thus adopted in the remainder of the paper. In the case of constant time-stepping, our methodology is essentially the same as the approach of Simo and Armero [15] and the TRLE algorithm described in [10, p. 163].

[1] This is the usual implementation of TR; see Gresho and Sani [9, p. 797]. An alternative interpretation (see Layton [10, p. 163]) is that an implicit midpoint evaluation of the the quadratic convection term $\vec{u}^{n+1/2}\cdot\nabla\,\vec{u}^{n+1/2}$ is computed via the second order update $(\vec{u}^{n+1}\cdot\nabla\,\vec{u}^{n+1} + \vec{u}^n\cdot\nabla\,\vec{u}^n)/2$.

Let $(\cdot,\cdot)$ denote the standard scalar or vector valued L^2 inner product defined on Ω. Given the velocity solution space $\mathcal{H}^1_{\vec{g}} = \{\,\vec{v}|\,\vec{v} \in H^1(\Omega)^2;\, \vec{v}|_{\Gamma_D} = \vec{g}\}$, the linearized semidiscrete problem can be formulated as a variational problem: given $(\,\vec{u}^n, p^n) \in \mathcal{H}^1_{\vec{g}^n} \times L^2(\Omega)$, we seek $(\,\vec{u}^{n+1}, p^{n+1}) \in \mathcal{H}^1_{\vec{g}^{n+1}} \times L^2(\Omega)$ such that

$$\frac{2}{k_{n+1}}(\,\vec{u}^{n+1},\,\vec{v}) + \nu\,(\nabla\,\vec{u}^{n+1}, \nabla\,\vec{v}) + (\,\vec{w}^{n+1}\cdot\nabla\,\vec{u}^{n+1},\,\vec{v}) - (p^{n+1}, \nabla\cdot\,\vec{v})$$

$$= \frac{2}{k_{n+1}}(\,\vec{u}^{n},\,\vec{v}) + \left(\frac{\partial\,\vec{u}^{\,n}}{\partial t},\,\vec{v}\right), \tag{2.7}$$

$$(\nabla\cdot\,\vec{u}^{n+1}, q) = 0 \tag{2.8}$$

for all $(\,\vec{v}, q) \in \mathcal{H}^1_{\vec{0}}(\Omega) \times L^2(\Omega)$.

Throughout this paper, spatial discretization will be done using a method-of-lines approach based on finite element approximation on a fixed spatial grid. Our algorithm methodology described below thus applies essentially verbatim to finite difference and finite volume discretizations. The domain Ω is split into finitely many nonoverlapping triangles τ, giving a triangulation $\mathcal{T}_h$. (The mesh parameter h can be associated with the length of the longest edge of a triangle from $\mathcal{T}_h$.) Low-order mixed approximation methods are not stable[2] in general. One mixed method that is stable is the so-called Taylor–Hood P_2–P_1 method, which uses continuous piecewise quadratic approximation for the velocity components and continuous piecewise linear approximation for pressure. We use Taylor–Hood approximation throughout this work but emphasize that the rapid convergence properties of the linear solver methodology described in the next section are essentially independent of the mixed approximation used.

Thus, using finite-dimensional approximation spaces $X \subset \mathcal{H}^1_{\vec{0}}$ and $M \subset L^2(\Omega)$, the fully discrete problem is to find $(\,\vec{u}_h^{n+1}, p_h^{n+1}) \in X_{\vec{g}} \times M$ such that

$$\frac{2}{k_{n+1}}(\,\vec{u}_h^{n+1},\,\vec{v}_h) + \nu\,(\nabla\,\vec{u}_h^{n+1}, \nabla\,\vec{v}_h) + (\,\vec{w}_h^{n+1}\cdot\nabla\,\vec{u}_h^{n+1},\,\vec{v}_h) - (p_h^{n+1}, \nabla\cdot\,\vec{v}_h)$$

$$= \frac{2}{k_{n+1}}(\,\vec{u}_h^{n},\,\vec{v}_h) + \left(\frac{\partial\,\vec{u}_h}{\partial t}^{\,n},\,\vec{v}_h\right), \tag{2.9}$$

$$(\nabla\cdot\,\vec{u}_h^{n+1}, q_h) = 0 \tag{2.10}$$

for all $(\,\vec{v}_h, q_h) \in X \times M$. The linear algebra version of (2.9)–(2.10) will be explicitly constructed in the next section.

Our adaptive time-stepping algorithm is a refined version of the "smart integrator" advocated by Gresho and Sani [9, sect. 3.16.4] and has three ingredients: time integration, the time-step selection method, and stabilization of the integrator. We briefly discuss each of these separately below so as to mirror the discussion in part I; see [8, sect. 1].

Time integration. We are conscious of the need to minimize potential round-off instability; thus our implementation of the TR–AB2 pair explicitly computes the discrete velocity updates scaled by the time step to avoid underflow and inhibit subtractive cancellation. Specifically, given $\vec{u}_h^n$, $\frac{\partial u_h^n}{\partial t}$, and the boundary update $\vec{g} := \frac{\vec{g}^{n+1} - \vec{g}^n}{k_{n+1}}$,

[2]See Elman, Silvester, and Wathen [5, Ch. 5] for a full discussion of inf-sup stability.

we first compute the pair $(\vec{d}_h^{\,n}, p_h^{n+1}) \in X_{\vec{g}} \times M$ such that

$$2(\vec{d}_h^{\,n}, \vec{v}_h) + \nu k_{n+1} (\nabla \vec{d}_h^{\,n}, \nabla \vec{v}_h) + k_{n+1}(\vec{w}_h^{n+1} \cdot \nabla \vec{d}_h^{\,n}, \vec{v}_h) - (p_h^{n+1}, \nabla \cdot \vec{v}_h)$$

$$= \left(\frac{\partial \vec{u}_h^{\,n}}{\partial t}, \vec{v}_h \right) - \nu (\nabla \vec{u}_h^{\,n}, \nabla \vec{v}_h) - (\vec{w}_h^{n+1} \cdot \nabla \vec{u}_h^{\,n}, \vec{v}_h), \tag{2.11}$$

$$(\nabla \cdot \vec{d}_h^{\,n+1}, q_h) = 0 \tag{2.12}$$

for all $(\vec{v}_h, q_h) \in X \times M$, and then we update the TR velocity field and the acceleration (time derivative of the velocity) via

$$\vec{u}_h^{n+1} = \vec{u}_h^{\,n} + k_{n+1} \vec{d}_h^{\,n}; \qquad \frac{\partial \vec{u}_h^{n+1}}{\partial t} = 2 \vec{d}_h^{\,n} - \frac{\partial \vec{u}_h^{\,n}}{\partial t}. \tag{2.13}$$

We will subsequently refer to (2.11)–(2.12) as the *discrete Oseen problem*. Note that the computed pressure field p_h^{n+1} is not needed for subsequent time steps and does not play a role in the time-step selection process described next.

Time-step selection. To control the time integration it is usual to place a user-specified *tolerance*, ε, on the L_2 norm of the truncation error at the next time step, $\vec{e}_h^{\,n+1}$, so that

$$\| \vec{e}_h^{\,n+1} \| \leq \varepsilon \| \vec{u}_h^{\,\infty} \|, \tag{2.14}$$

where $\| \vec{u}_h^{\,\infty} \|$ is (a possibly user-specified estimate of) the maximum norm of the method-of-lines solution over the prescribed time interval.[3] Assuming that the underlying ODE system has smooth third derivatives in time (so that the TR time integration is indeed second-order accurate) standard manipulation of Taylor series shows that the ratio of successive truncation errors is proportional to the cube of the ratio of successive time steps. This motivates the following time-step selection heuristic:

$$k_{n+2} = k_{n+1} \left(\varepsilon / \| \vec{e}_h^{\,n+1} \| \right)^{\frac{1}{3}}. \tag{2.15}$$

The local truncation error $\vec{e}_h^{\,n+1}$ is estimated by comparing the TR velocity solution $\vec{u}_h^{\,n+1}$ with the AB2 velocity solution $\vec{u}_*^{\,n+1}$ computed using the explicit update formula

$$\vec{u}_*^{\,n+1} = \vec{u}_h^{\,n} + \frac{k_{n+1}}{2} \left[\left(2 + \frac{k_{n+1}}{k_n} \right) \frac{\partial u_h^n}{\partial t} - \left(\frac{k_{n+1}}{k_n} \right) \frac{\partial \vec{u}_h^{\,n-1}}{\partial t} \right], \tag{2.16}$$

using the standard estimate (cf. part I, [8, p. 2021])

$$\vec{e}_h^{\,n+1} = (\vec{u}_h^{\,n+1} - \vec{u}_*^{\,n+1}) / [3(1 + k_n / k_{n+1})]. \tag{2.17}$$

To implement this methodology in a practical code there are two start-up issues that need to be addressed:

1. AB2 is not self-starting. To start the simulation we require a finite element function $\vec{u}_h^{\,0}$ with boundary data $\vec{g}^{\,0}$ that satisfies the discrete incompressibility constraint

$$(\nabla \cdot \vec{u}_h^{\,0}, q_h) = 0 \quad \text{for all } q_h \in M. \tag{2.18}$$

[3] $\| \vec{u}_h^{\,\infty} \| = 1$ in all of the examples discussed in this paper.

The initial acceleration (and concomitant pressure) is then computed by solving the discrete (potential flow) problem: given the boundary update $\vec{g} := \frac{\vec{g}^1 - \vec{g}^0}{k_1}$, we compute the pair $(\frac{\partial \vec{u}_h^0}{\partial t}, p_h^0) \in X_{\vec{g}} \times M$ such that

$$\left(\frac{\partial \vec{u}_h^0}{\partial t}, \vec{v}_h\right) - (p_h^0, \nabla \cdot \vec{v}_h) = -\nu\,(\nabla \vec{u}^0, \nabla \vec{v}) - (\vec{u}^0 \cdot \nabla \vec{u}^0, \vec{v}), \tag{2.19}$$

$$\left(\nabla \cdot \frac{\partial \vec{u}_h^0}{\partial t}, q_h\right) = 0 \tag{2.20}$$

for all $(\vec{v}_h, q_h) \in X \times M$. We then set $n = 0$ and define $\vec{w}_h^1 = \vec{u}_h^0 + k_1 \frac{\partial \vec{u}_h^0}{\partial t}$ so as to construct the discrete Oseen problem (2.9)–(2.10). Solving this discrete Oseen problem gives $(\vec{u}_h^1, p_h^1)$ (the TR velocity and pressure) at the end of the first time step. The acceleration at time $t = k_1$ is then computed using $\frac{\partial \vec{u}_h^1}{\partial t} = \frac{2}{k_1}(\vec{u}_h^1 - \vec{u}_h^0) - \frac{\partial \vec{u}_h^0}{\partial t}$ and allows us to compute the AB2 velocity at the second time step. To complete the start-up process, time step control is then switched on at the third time step ($k_1 = k_0$).

2. Choice of initial time step k_0. Several strategies are available with which to start the TR method. Our strategy is to select a conservatively small value for k_0 (say, 10^{-8}). With such a choice we will have rapid growth in the time step: typically $\|\vec{e}_h^n\| = O(\texttt{eps})$ for the first few time steps (where `eps` is machine precision), and so $k_{n+1}/k_n = O((\varepsilon/\texttt{eps})^{1/3}) \approx 10^4$ when $\varepsilon = 10^{-4}$. This rapid growth implies that, for small values of n, we see exponential growth in the time step, and with very few such steps (typically 2–4) a time step is obtained that is commensurate with the "initial response time." See part I [8, p. 2021] for further discussion of this point.

A general purpose ODE code in a software library will typically have multiple bells and whistles. In contrast, our time-stepping algorithm has just one "trip":

1. Time-step rejection. After computing the new time step via (2.15), we check to see if the next step is seriously reduced,[4] i.e., $k_{n+2} < 0.7k_{n+1}$, or equivalently that $\|\vec{e}_h^{n+1}\| > (1/0.7)^3\varepsilon$. If this happens, then the next time step is rejected: the value of k_{n+1} is multiplied by $(\varepsilon/\|\vec{e}_h^{n+1}\|)^{1/3}$, and the current step is repeated with this smaller value of k_{n+1}.

Stabilization of the integrator. As discussed earlier, the TR method is prone to "ringing" when solving stiff problems (typically for PDEs when using very small spatial grid sizes to resolve fine detail) with relatively large tolerances on the time step or toward the end of a simulation when close to steady state. Situations such as these are discussed by Osterby [11] along with a variety of means of suppressing the oscillations. Our code implements an alternative strategy—*time-step averaging*. The averaging is invoked periodically every n_* steps. For such a step, we save the values $t_* = t_n$ and $\vec{u}_h^* = \vec{u}_h^n$, and, having computed the TR update $\vec{d}_h^n$ via (2.11)–(2.12), we set $t_n = t_{n-1} + \frac{1}{2}k_n$ and $t_{n+1} = t_* + \frac{1}{2}k_{n+1}$ and define the new "shifted" solution

[4] For example, if the iterative solver discussed in section 3 does not solve the discrete Oseen problem to the required accuracy, then $\|\vec{e}_h^n\|$ will be larger than we would expect for the current time step. If the step is repeated with a smaller step size, then the associated linear algebra problem is more easily solved so the time-stepping algorithm can recover.

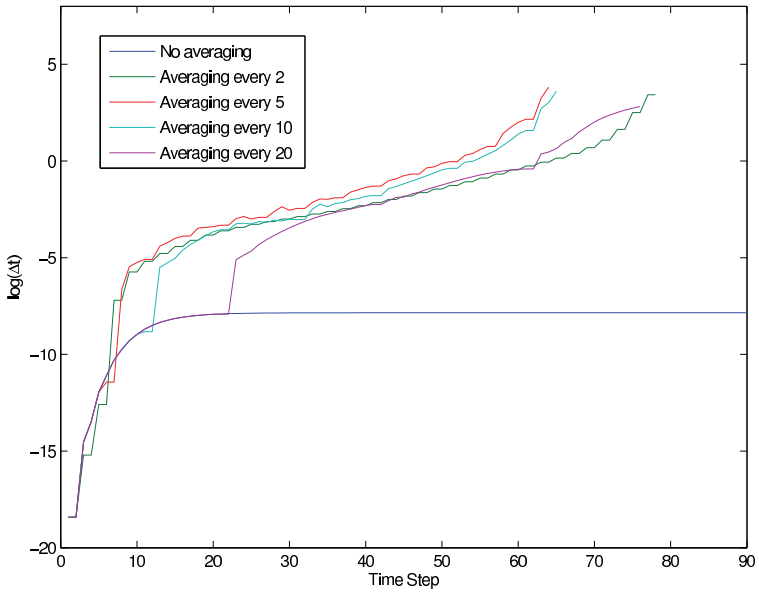

FIG. 1. *Stabilized TR–AB2 integrator with periodic averaging:* $\log k_j$ *versus time-step number* j *for a driven cavity flow with viscosity* $\nu = 1/100$ *being spun up from rest.*

vectors so that

$$\vec{u}_h^{\,n} = \frac{1}{2}(\,\vec{u}_h^{\,*} + \,\vec{u}_h^{\,n-1}), \qquad \frac{\partial\,\vec{u}_h^{\,n}}{\partial t} = \frac{1}{2}\left(\frac{\partial\,\vec{u}_h^{\,n}}{\partial t} + \frac{\partial\,\vec{u}_h^{\,n-1}}{\partial t}\right), \tag{2.21}$$

$$\vec{u}_h^{\,n+1} = \,\vec{u}_h^{\,*} + \tfrac{1}{2}k_{n+1}\,\vec{d}_h^{\,n}, \qquad \frac{\partial\,\vec{u}_h^{\,n+1}}{\partial t} = \,\vec{d}_h^{\,n}. \tag{2.22}$$

We then compute the next time step using (2.15) and continue the integration. The averaging process annihilates any contribution of the form $(-1)^n$ to the solution and its time derivative, thus cutting short the "ringing" while maintaining second-order accuracy. In our code the parameter n_* is a fixed parameter—typically 10. (A way of calculating a suitable value n_* on the fly is discussed in part I [8, p. 2023].) The benefit of this simple stabilization strategy is illustrated in Figure 1, which shows the behavior of stabilized and unstabilized TR–AB2 for a driven cavity flow problem for a Reynolds number $Re = UL/\nu = 100$. The fluid is initially at rest, and the tangential velocity of one of the boundaries is smoothly increased to a value of unity; full details are discussed later. Since the underlying Reynolds number is small enough, the flow solution tends to a steady state as $t \to \infty$.

Looking at Figure 1, we see that the unstabilized TR–AB2 integrator generates a constant time step as the steady state is approached. This behavior is erroneous in the sense that if we were to follow the physics, then the time step would increase as we approach the steady state. This is what we see when we stabilize the integrator, and it is independent of the frequency of averaging. Note that there is a drop-off in performance when we average too frequently or too infrequently—our default choice of $n_* = 10$ is essentially a compromise between enforcing stability and maintaining accuracy. This is the value of n_* that was used when generating the results presented in section 4.

3. Solving the discrete Oseen system. Let $\{\phi_i\}_{i=1}^{n_u}$ define the basis set for the approximation of a function from the space $H_0^1 := \{v | v \in H^1(\Omega); v|_{\Gamma_D} = 0\}$, and let $\{\psi_j\}_{j=1}^{n_p}$ define a basis set for the discrete pressure. The fully discrete solution $(\vec{u}_h^{n+1}, p_h^{n+1})$ corresponding to the Oseen problem (2.9)–(2.10) is given by the expansions

$$\vec{u}_h^{n+1} = \left[\sum_{i=1}^{n_u} \alpha_i^{x,n+1}\phi_i, \sum_{i=1}^{n_u} \alpha_i^{y,n+1}\phi_i\right] + \vec{g}^{n+1}, \quad p_h^{n+1} = \sum_{k=1}^{n_p} \alpha_k^{p,n+1}\psi_k, \tag{3.1}$$

where $\boldsymbol{\alpha}^{x,n+1}$, $\boldsymbol{\alpha}^{y,n+1}$, and $\boldsymbol{\alpha}^{p,n+1}$ represent vectors of coefficients. These are computed by solving the linear equation system defined below.

Given the velocity basis set, we define so-called velocity matrices M_v, A_v, and N_v, representing identity, diffusion, and convection operators in the velocity space, respectively:

$$M_v = [M_v]_{ij} = (\phi_i, \phi_j), \tag{3.2}$$

$$A_v = [A_v]_{ij} = (\nabla\phi_i, \nabla\phi_j), \tag{3.3}$$

$$N_v(\vec{u}_h) = [N_v]_{ij} = (\vec{u}_h \cdot \nabla\phi_i, \phi_j). \tag{3.4}$$

Combining the three velocity matrices and using the linearization in (2.7) defines the velocity convection-diffusion matrix at time t_{n+1}:

$$F_v^{n+1} := \frac{1}{k_{n+1}} M_v + \nu A_v + N_v(\vec{w}_h^{n+1}), \tag{3.5}$$

with $\vec{w}_h^{n+1} = (1 + (k_{n+1}/k_n))\,\vec{u}_h^n - (k_{n+1}/k_n)\,\vec{u}_h^{n-1}$. In addition, given the pressure basis set, we can define a discrete divergence matrix $B = [B_x, B_y]$ via

$$B_x = [B_x]_{ki} = -\left(\psi_k, \frac{\partial\phi_i}{\partial x}\right), \tag{3.6}$$

$$B_y = [B_y]_{ki} = -\left(\psi_k, \frac{\partial\phi_i}{\partial y}\right). \tag{3.7}$$

Looking ahead to preconditioning the discrete system, we also define pressure matrix analogues of M_v, A_v, and N_v, representing identity, diffusion, and convection operators in the pressure space:

$$M_p = [M_p]_{k\ell} = (\psi_k, \psi_\ell), \tag{3.8}$$

$$A_p = [A_p]_{k\ell} = (\nabla\psi_k, \nabla\psi_\ell), \tag{3.9}$$

$$N_p(\vec{w}_h) = [N_p]_{k\ell} = (\vec{w}_h \cdot \nabla\psi_k, \psi_\ell). \tag{3.10}$$

Finally, using the definitions (3.1)–(3.7), the discretized Oseen problem can be expressed as the following system: find $[\boldsymbol{\alpha}^{x,n+1}, \boldsymbol{\alpha}^{y,n+1}, \boldsymbol{\alpha}^{p,n+1}] \in \mathbb{R}^{n_u \times n_u \times n_p}$ such that

$$\begin{bmatrix} F_v^{n+1} & 0 & B_x^T \\ 0 & F_v^{n+1} & B_y^T \\ B_x & B_y & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{\alpha}^{x,n+1} \\ \boldsymbol{\alpha}^{y,n+1} \\ \boldsymbol{\alpha}^{p,n+1} \end{bmatrix} = \begin{bmatrix} \mathbf{f}^{x,n+1} \\ \mathbf{f}^{y,n+1} \\ \mathbf{f}^{p,n+1} \end{bmatrix}. \tag{3.11}$$

The right-hand-side vector $\mathbf{f}$ is constructed from the boundary data $\vec{g}^{n+1}$, the computed velocity $\vec{u}_h^n$ at the previous time level, and the acceleration $\frac{\partial \vec{u}_h^n}{\partial t}$.

The coefficient matrix in (3.11) may be written in the equivalent form

$$K := \begin{bmatrix} \mathcal{F}_v^{n+1} & B^T \\ B & 0 \end{bmatrix}, \tag{3.12}$$

where $\mathcal{F}_v^{n+1}$ is a 2×2 block diagonal matrix, with diagonal blocks F_v^{n+1} defined as in (3.5). Thus, at every time level, we are faced with the task of solving a square nonsingular linear equation system $K\boldsymbol{\alpha} = \mathbf{f}$. This is done using a (right-) preconditioned Krylov subspace method. Such methods start with some guess $\boldsymbol{\alpha}_0$, with residual $\mathbf{r}_0 = \mathbf{f} - K\boldsymbol{\alpha}_0$; given a preconditioner, P, say, to be defined later, construct a sequence of approximate solutions of the form

$$\boldsymbol{\alpha}_k = \boldsymbol{\alpha}_0 + \mathbf{p}_k, \tag{3.13}$$

where $\mathbf{p}_k$ is a vector in the k-dimensional *Krylov space*

$$\mathcal{K}_k(\mathbf{r}_0, KP^{-1}) = \text{span}\{\mathbf{r}_0, KP^{-1}\mathbf{r}_0, \ldots, (KP^{-1})^{k-1}\mathbf{r}_0\}. \tag{3.14}$$

The preconditioned GMRES method is used herein. A feature of GMRES is that it is an optimal Krylov solver in that it computes the unique iterate of the form (3.13) for which the Euclidean (or root mean square) norm of the residual vector is smallest—the classical convergence estimate is

$$\|\mathbf{r}_k\|_2 = \min_{\phi_k(0)=1} \|\phi_k(KP^{-1})\mathbf{r}_0\|_2,$$

where ϕ_k is the set of polynomials of degree k. Step m of the process requires one matrix-vector product together with a set of m vector operations, making its cost, in terms of both operation counts and storage, proportional to mn, where $n = 2n_u + n_p$ is the dimension of the system (3.11). A full discussion of GMRES convergence properties can be found in [5, Ch. 4], together with details of the construction of successive GMRES iterates.

At every time level, we set the initial vector $\boldsymbol{\alpha}_0 = \mathbf{0}$ and run the preconditioned GMRES process until either a fixed number of iterations (`maxit`) is reached, or else the stopping-test

$$\frac{\|\mathbf{f} - K\boldsymbol{\alpha}_m\|_2}{\|\mathbf{r}_0\|_2} < \texttt{itol}$$

is satisfied. We denote the solver strategy by GMRES(`maxit`, `itol`). Typically, we set `maxit` to 50, and `itol` to 10^{-6}. The big task is to construct a preconditioner P such that the stopping-test is satisfied for small m. Furthermore, we would like this m to be independent of the discretization parameter h and the viscosity ν. Given that a complete description of our preconditioning methodology is given in [13] and [5, Ch. 8], we simply outline the key features here.

The general form of an ideal preconditioner is

$$P = \begin{pmatrix} \mathcal{F}_v^{n+1} & B^T \\ 0 & -X \end{pmatrix}, \tag{3.15}$$

where the $n_p \times n_p$ matrix X is an approximation to the pressure Schur complement matrix $S = B(\mathcal{F}_v^{n+1})^{-1}B^T$. We note that if the exact Schur complement $X := S$ were used in (3.15), then GMRES would give the exact solution in two iterations, that is,

$\boldsymbol{\alpha}_2 = \boldsymbol{\alpha}$; see Murphy, Golub, and Wathen [7]. Since the Schur complement matrix S is dense, then an equally effective yet relatively inexpensive approximation to it is needed if this approach is to be practical. Such an approximation is that developed by Kay, Loghin, and Wathen [6] and is referred to herein as *pressure convection-diffusion preconditioning*. It is given by setting $X = A_p(F_p^{n+1})^{-1}M_p$, with the pressure matrix operators A_p and M_p given in (3.9) and (3.8), respectively, and

$$F_p^{n+1} = \frac{1}{k_{n+1}}M_p + \nu A_p + N_p(\vec{w}_h^{n+1}), \tag{3.16}$$

defining the pressure space analogue of the F_v^{n+1} operator in (3.5). The properties of this Schur complement approximation are the subject of ongoing analysis; see [3] for some theoretical results in the steady-state case. Numerical experiments showing the good performance of this preconditioning strategy in the context of steady-state flow problems are given in [16] and [2].

Note that preconditioning with P requires the action of the inverse of $\mathcal{F}_v^{n+1}$ and X at each GMRES iteration:

$$P^{-1} = \begin{pmatrix} (\mathcal{F}_v^{n+1})^{-1} & (\mathcal{F}_v^{n+1})^{-1}B^T X^{-1} \\ 0 & -X^{-1} \end{pmatrix}. \tag{3.17}$$

Using the pressure convection-diffusion approximation to X, we see that

$$X^{-1} = M_p^{-1}(F_p^{n+1})A_p^{-1}, \tag{3.18}$$

and thus preconditioning is done by effecting the action of the inverse operators $(\mathcal{F}_v^{n+1})^{-1}$, A_p^{-1}, and M_p^{-1}. In practice, these matrix operations can be done very efficiently using algebraic multigrid (AMG). More specifically, all the results in the next section are computed using an *inexact* preconditioner where the actions of $(F_v^{n+1})^{-1}$ and A_p^{-1} are approximated by two AMG V-cycles and the action of the inverse mass matrix M_p^{-1} is approximated by five iterations of a diagonally scaled conjugate gradient algorithm.[5]

The AMG code that we use for this is a MATLAB version of the subroutine `HSL_MI20` [4]. It should be stressed that we use this subroutine as a *black-box*—we specify three (point Gauss–Seidel) smoothing sweeps (no special reordering) at each level, and all AMG coarsening parameters are set to the default values. The realization that we were able to generate results without having to incorporate streamline diffusion into the preconditioner was a big surprise for us.[6] The fact that we are using standard Galerkin approximation without "tuning parameters" makes for a very clean discretization and, in our opinion, makes our methodology look very attractive.

4. Numerical results. The first model problem is the classical lid-driven cavity. The motivation for considering this is to demonstrate the effectiveness of our solver when it is used to time step to a steady state. The second model problem is another well-studied problem, namely, that of a channel flow with a cylindrical obstruction. The version of the model that we consider is that proposed by Dettmer and Perić in [1]. An alternative problem statement and a benchmark solution is given by Turek [17, Ch. 1]. The motivation for studying this problem is to show that our solver can be

[5] Diagonally scaling the mass matrix gives a perfectly conditioned operator; see [5, Lemma 6.3].

[6] The mass matrix contribution coming from the time-stepping is the crucial ingredient here—as the temporal error tolerance is reduced, the effectiveness of the AMG solver is increased.

used to "follow the physics" of a transition of a flow from a state of rest to a periodic state of vortex shedding. For a related and detailed analysis of the start-up flow past a cylinder, see Gresho and Sani [9, Sec. 3.19].

Example 4.1. Consider a spatially discretized system (2.7)–(2.8) defined on a unit square cavity domain. The initial condition is $\vec{u}_0(\vec{x}) = 0$ in the cavity, and an enclosed flow boundary condition $\vec{g} = \vec{0}$ is imposed on three of the walls together with a time-dependent velocity $\vec{g} = (1 - e^{-5t}, 0)$ on the top boundary $0 < x < 1; y = 1$. This models a slow start-up from rest. For sufficiently small Reynolds number $Re = 1/\nu < 13,000$ the flow tends to a steady state. This consists of a clockwise rotating primary flow, secondary recirculation regions in the two bottom corners, and a tertiary recirculation on the left-hand wall; see Shankar and Deshpande [12]. For larger values of the Reynolds number the steady flow is not stable. At a very large Reynolds number the flow will be chaotic and turbulent.

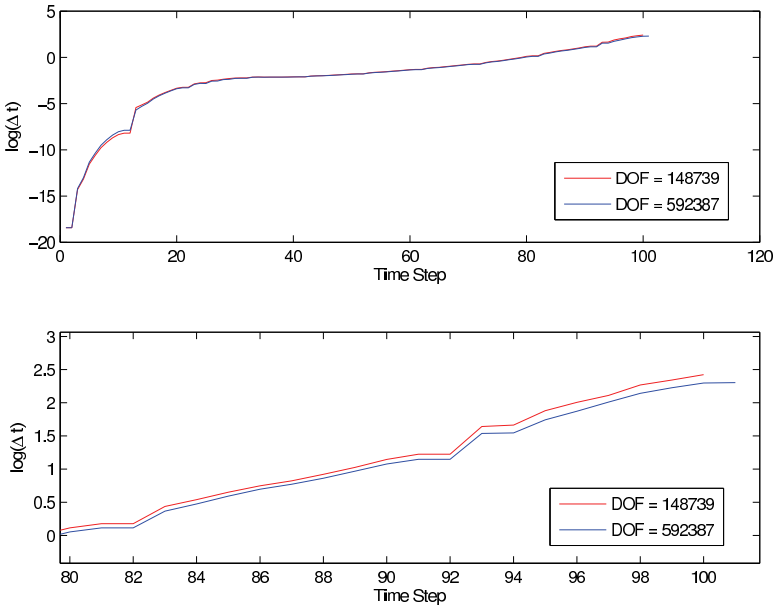

FIG. 2. *Stabilized TR–AB2 integrator with periodic averaging for Example* 4.1 *with viscosity* $\nu = 1/1000$*:* log k_j *versus time step number* j*. Top: Entire evolution. Bottom: Zoom showing the almost perfect agreement over the final* 20 *time steps.*

Figure 2 shows the evolution of the time step when we run our solver for a problem with $\nu = 1/1000$ with time-stepping tolerance $\varepsilon = 10^{-4}$. For simplicity, we use a uniform mesh of right-angled triangles with edge length h, and we compare the behavior on two meshes—a basic mesh (red curve) with $1/h = 64$ and a refined mesh (blue curve) with $1/h = 128$. The time integration was run to a final time of 100 time units. Using either mesh, the time step grows monotonically with time and ultimately reaches a value of $O(10)$ time units. The time-step evolution can also be seen to be essentially independent of the spatial subdivision—this suggests that the simulations are time-accurate. The time-step behavior can also be seen to be consistent with that in Figure 1 with $n_* = 10$, which is for an order of magnitude larger viscosity.

The performance of the preconditioned GMRES(50, 10^{-6}) solver is plotted in Figure 3. We observe that the discrete Oseen system becomes progressively more difficult to solve as the time step grows. On a positive note, however, we also see

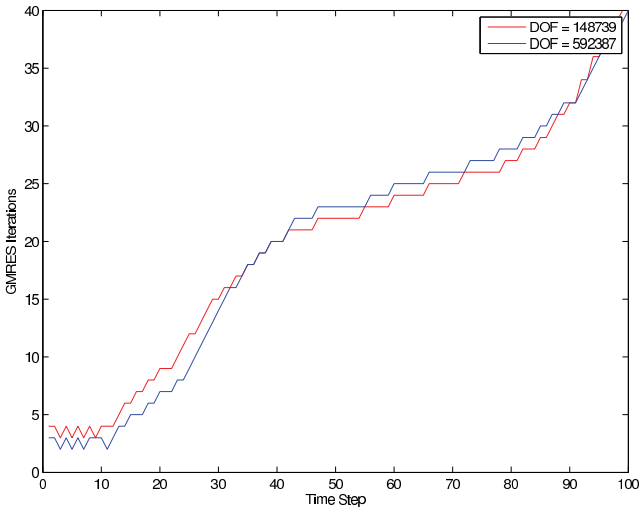

FIG. 3. *Stabilized TR–AB2 integrator with periodic averaging for Example* 4.1 *with viscosity* $\nu = 1/1000$*: Number of GMRES iterations versus time step number* j.

that the solver performance at each time step is completely insensitive to the mesh refinement level!

Example 4.2. Consider a spatially discretized system (2.7)–(2.8) defined on a rectangular domain $-5 < x < 16, -4.6 < y < 4.5$ with a cylindrical obstruction of diameter unity centered at the point $(0, 0)$. The initial condition is $\vec{u}_0(\vec{x}) = 0$ in the domain, and a zero flow boundary condition is imposed on the cylinder boundary together with a time-dependent velocity $\vec{g} = (1 - e^{-5t}, 0)$ on the left-hand (inflow) boundary, as well as at the top and the bottom of the channel. The right-hand boundary $x = 16, -4.6 < y < 4.5$ satisfies the natural outflow condition (1.4) for all time. For a viscosity coefficient in the range $1/1000 \leq \nu \leq 1/100$ the flow tends to a time periodic vortex shedding solution which subjects the cylinder to oscillating lift and drag forces acting parallel and perpendicular to the direction of the flow, respectively.

The results in Figure 4 illustrate what happens when we run our solver on the problem with $\nu = 1/100$, using the relatively coarse mesh shown in Figure 5, with the time-stepping tolerance $\varepsilon = 10^{-4}$. The top plot shows a snapshot of the flow solution during the shedding cycle. Also shown is the evolution of the lift coefficient from the rest state, and the cyclic variation of the drag coefficient after shedding has been established. With this accuracy tolerance the algorithm generates a constant time step of 0.05 once in the shedding regime—this corresponds to approximately 100 sample points per shedding cycle. Looking at the time-step evolution in Figure 4, we can identify four distinct phases. First, we have a "fake" initial transient which lasts 1–5 time units and which is associated with the dynamic boundary condition. Note that there is a noticeable "judder" at about 5 time units—roughly speaking when the initial influx hits the cylinder. Between 5 and 35 time units the integrator runs with a constant time step of about 0.13, corresponding to the lengthening of the pair of recirculating eddies in the wake of the cylinder. Finally, after approximately 35 time units, stability is lost, the flow symmetry is broken, and the time step automatically cuts back, ultimately settling on the constant value that is appropriate for the accurate computation of the vortex shedding.

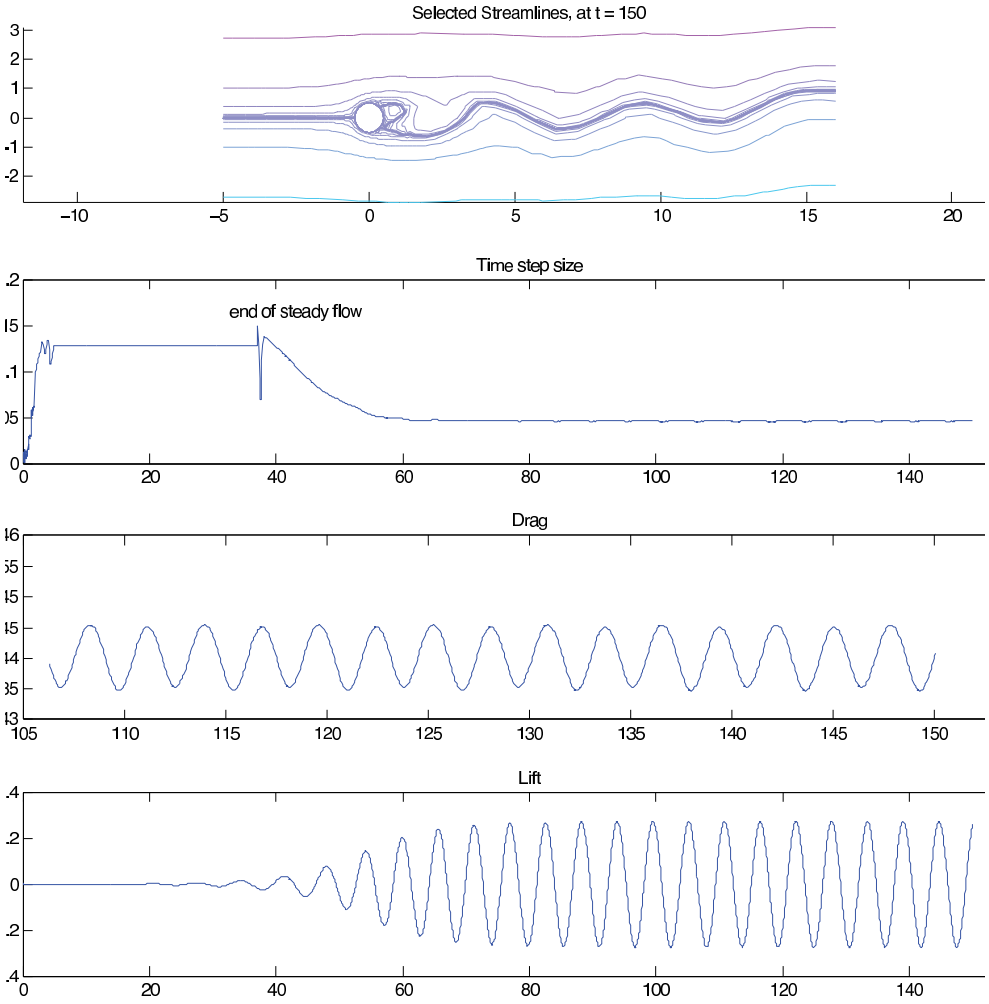

FIG. 4. *Solution data for Example* 4.2 *for* $\nu = 1/100$ *with accuracy* $\varepsilon = 10^{-4}$. *The y-axis limits for the time step in the second plot range from* 0 *to* 0.2, *and those for the drag coefficients in the third plot range from* 1.43 *to* 1.46.

In the remainder of the paper we will critically assess the performance of our solver, first when run with $\nu = 1/100$, and second when run with $\nu = 1/400$. Our results can then be directly compared with those in [1, pp. 1213–1221], where the same problems are solved using fixed time-stepping, together with fully implicit approximation (that is, solving nonlinear equations using Newton iteration at each time level). We solve both of these flow problems on three meshes. The coarsest mesh, so-called level 1, is illustrated in Figure 5. It contains 999 triangles and has $n_u = 2055$ velocity degrees of freedom. An intermediate mesh, so-called level 2, is obtained by a uniform refinement of the coarsest mesh and contains 3996 triangles and has $n_u = 8106$ velocity degrees of freedom. Note that the newly introduced nodes are "moved" to the cylinder boundary to give a better resolution of the circular geometry. The finest mesh, so-called level 3, is obtained by a uniform refinement of the intermediate mesh, but this time excluding the two regions at the top and the bottom that adjoin the

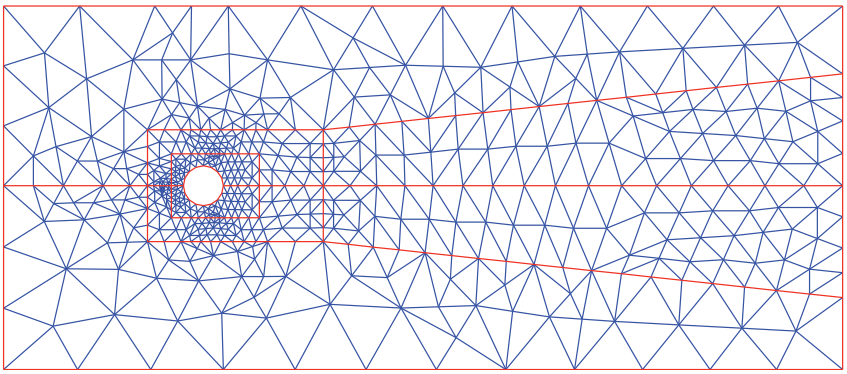

FIG. 5. *Reference mesh for Example* 4.2.

inflow in Figure 5. This gives a mesh of 12728 triangles with $n_u = 25101$ velocity degrees of freedom.

We consider the easier case of $\nu = 1/100$ first. In Figure 6 we present a comparison plot showing the evolution of the lift coefficient with different mesh refinement levels. We note that the results computed on the level 2 and level 3 meshes are essentially indistinguishable from one another. Note that the frequency of shedding is a major point of physical interest. When we compare these results with the lift coefficient evolution computed using a fully implicit second-order time-stepping method that is given in [1, Fig. 15], we find very a close match—the frequency of shedding and the amplitude of the lift coefficient are identical in the eyeball norm.

The evolution of the associated time step is also shown in Figure 6. We note first that, independent of the spatial refinement, the character of the time-step evolution shows the four distinct phases exhibited in Figure 4. We can also see that the constant time step that is used in the second phase (the development of the recirculation zones in the cylinder wake) is mesh-dependent. The fact that the time step that is used in the shedding phase is essentially mesh-*independent* (certainly on the two finer levels) confirms that the integrator is time-accurate in this regime. Also shown in Figure 6 is the comparison of the iterative solver performance for the three refinement levels. From this, we see that our preconditioning strategy is extremely efficient—typically taking 23 GMRES iterations per time step in the shedding phase on the intermediate mesh and 29 GMRES iterations per time step on the most refined mesh. We can also see that the algorithm is robust (in the sense that there are few rejected steps) in the "tough" phases when the solver fails to converge to the accuracy tolerance in `maxit` iterations—for example, during the second phase of the flow evolution when using the most refined mesh. Our interpretation of this situation is that the time step that is generated by the crude $\varepsilon = 10^{-4}$ tolerance is probably overly optimistic during the second phase of the flow evolution. This reduces the effectiveness of the algebraic multigrid preconditioner[7] for the velocity convection-diffusion operator $(F_v^{n+1})^{-1}$. Evidence for this comes from solving the same problem on the intermediate mesh with a tighter error tolerance of $\varepsilon = 10^{-7}$. This generates a smooth hump-shaped time-step evolution with a maximum time step of 0.03 at about 17.5 time units and gives constant time steps in the shedding phase which are an order

[7] We would get much better robustness if we followed the recipe in [5, p. 361] and added a streamline diffusion stabilization term to the discrete operator prior to the set-up phase of the AMG solver.

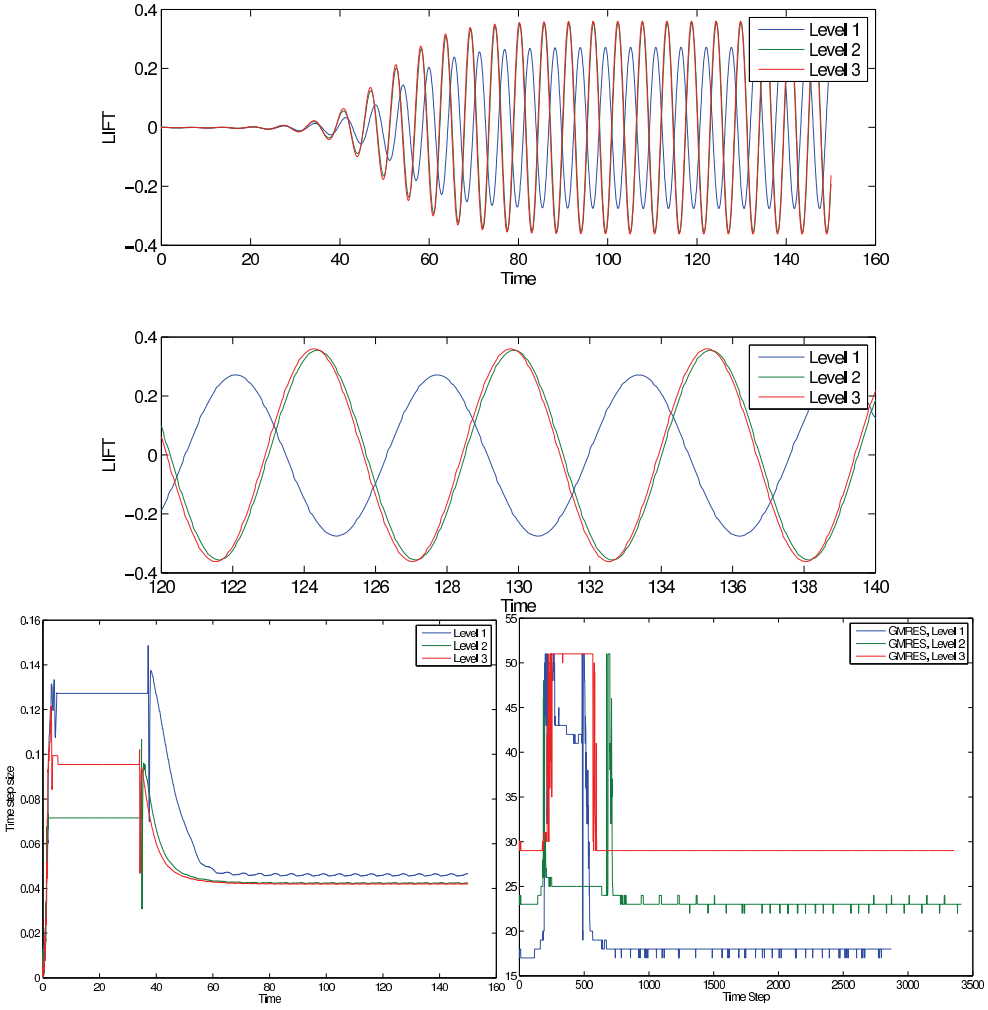

FIG. 6. *Top: Lift coefficient evolution for Example* 4.2 *for* $\nu = 1/100$ *with accuracy* $\varepsilon = 10^{-4}$. *Bottom: Time step size and number of GMRES iterations versus time step.*

of magnitude smaller than those in Figure 6. In this case, the linear solver has no problem at all—GMRES always satisfies the `itol`$=10^{-6}$ residual tolerance criterion in fewer than 25 iterations.

In these experiments the accuracy of the linearized TR–AB2 algorithm is almost identical to a fully implicit version of the TR–AB2 algorithm. To give a specific example, the shedding computation on the intermediate mesh was recomputed using a fully implicit version of TR. Thus, at every time step we iterated the Oseen solve (2.11)–(2.12) with a simple fixed point (Picard) iteration. The initial approximation for the convective velocity $(\vec{w}_h^{n+1})^{(0)}$ was given by $\vec{u}_h^n$. At the kth step we set up the system (2.11)–(2.12) with a convection field $(\vec{w}_h^{n+1})^{(k)}$ and solved it using preconditioned GMRES to an accuracy of `itol`$=10^{-6}$ to get the solution $(d_h^{n+1})^{(k)}$. This in turn gives the updated velocity approximation $(\vec{w}_h^{n+1})^{(k+1)} := (w_h^{n+1})^{(k)} + k_{n+1}(d_h^{n+1})^{(k)}$. Our fixed point iteration process was terminated when the norm of the difference between successive velocity iterates was less than 10^{-3}. Typically, this led to two or

three Picard iterations at every time step. The time step sequence that resulted using this nonlinear approach turned out to be almost identical to that generated by our linearized method. The complete history and a zoom in the shedding regime are shown in Figure 7. Regarding the relative accuracy of the corresponding flow solutions, the shedding frequencies are essentially identical, and the amplitude of the lift coefficient is slightly bigger, i.e., more accurate, in the nonlinear case. The total computation time was almost exactly three times longer in the nonlinear case, however!

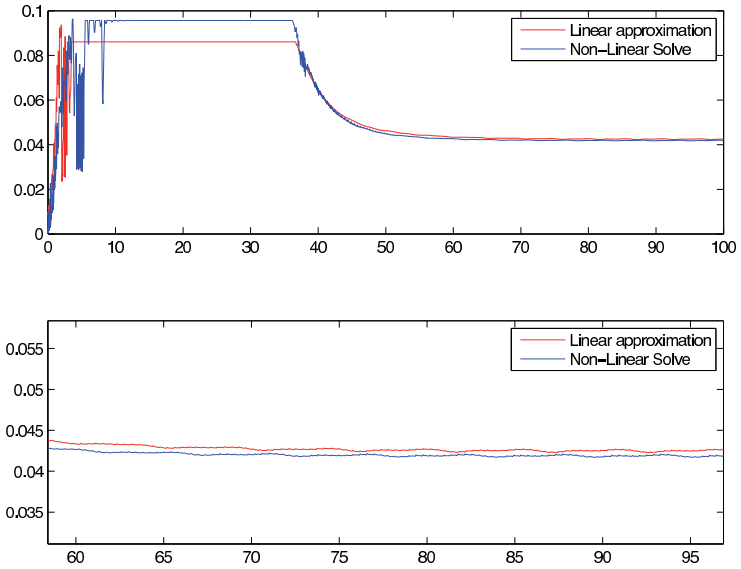

FIG. 7. *Top: Linearized versus nonlinear time stepping for Example* 4.2 *for* $\nu = 1/100$ *with accuracy* $\varepsilon = 10^{-4}$. *Bottom: Zoom of the time step size versus time in the shedding regime.*

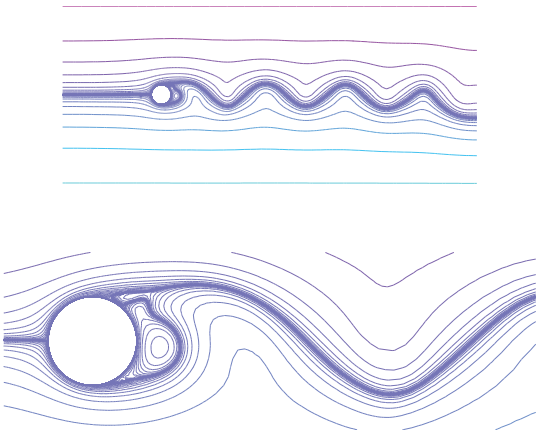

FIG. 8. *Solution snapshot for Example* 4.2 *for* $\nu = 1/400$ *with accuracy* $\varepsilon = 10^{-4}$.

We conclude with a discussion of the more challenging case of $\nu = 1/400$. A snapshot of the flow solution computed on the finest mesh is illustrated in Figure 8. Figure 9 shows a comparison plot showing the evolution of the lift coefficient with the three different meshes. We note that the results computed on the level 2 and level 3

meshes are in close agreement. For either mesh the amplitude of the lift coefficient in the shedding phase is clearly greater than 1.1. This makes the accuracy comparable to the benchmark results obtained using a fully implicit second-order method with "very small time steps," quoting from the legend in Figure 23 in [1].

Looking closely at the time-step evolution, we find that the time step tends to a value of about 0.02 time units in the shedding phase, independent of the mesh. We also see that the time step is repeatedly cut back on the refined mesh because the GMRES solver is not able to meet the tolerance in `maxit` iterations. In our view, the fact that we are able to generate a solution to this flow problem shows the inherent robustness of our solver methodology. To generate a perfectly smooth time-step evolution and simultaneously keep the GMRES iteration counts under control would require us to rerun the fine grid computation using a much tighter time accuracy tolerance, say, $\varepsilon = 10^{-7}$.

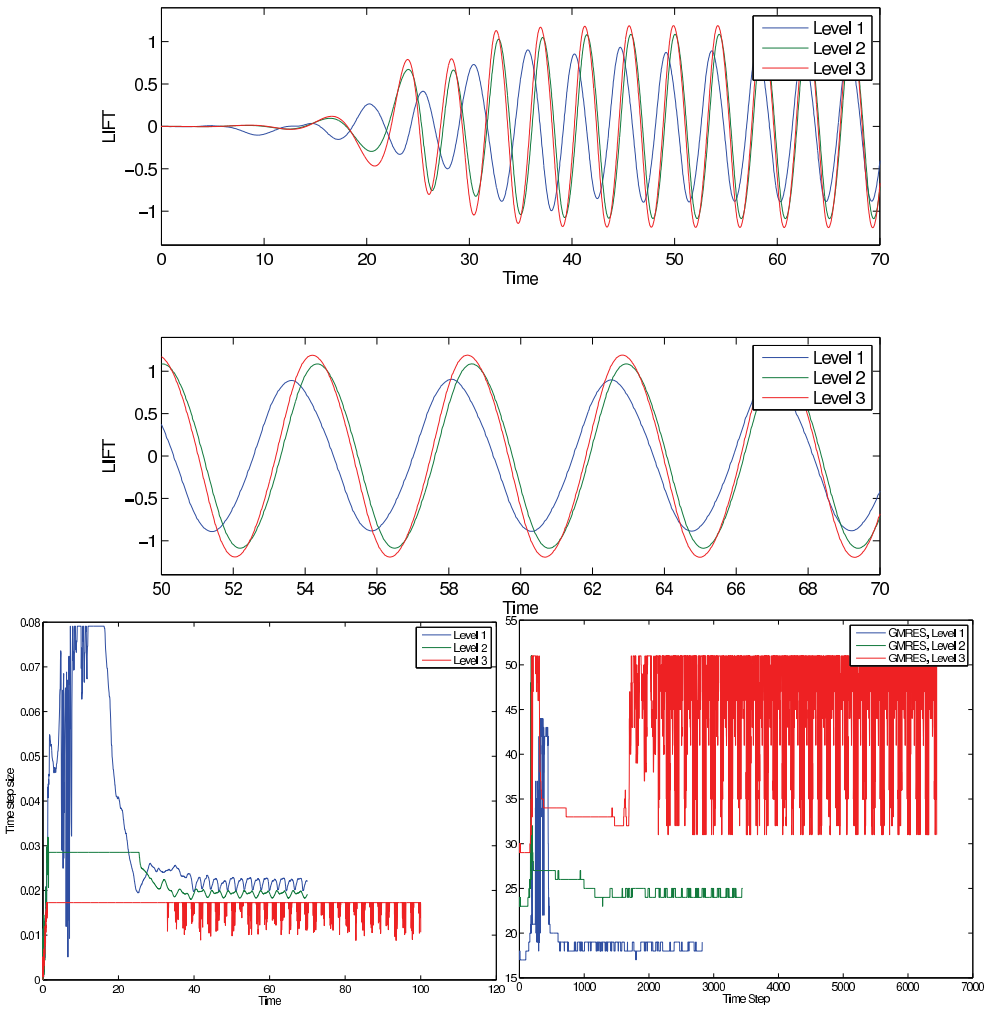

FIG. 9. *Top: Lift coefficient evolution for Example* 4.2 *for* $\nu = 1/400$ *with accuracy* $\varepsilon = 10^{-4}$. *Bottom: Time step size and number of GMRES iterations versus time step.*

5. Concluding remarks. Our numerical experiments show that even simple flow problems can have quite complex time scales, some physical and some of numerical origin. It is clear that some form of adaptive time integrator is essential in order to efficiently respond to the different time scales, and, given the wide range of dynamics taking place during these simulations, it is rather reassuring to see the TR–AB2 integrator find the appropriate time step during all the phases.

REFERENCES

[1] W. Dettmer and D. Perić, *An analysis of the time integration algorithms for the finite element solutions of incompressible Navier–Stokes equations based on a stabilised formulation*, Comput. Methods Appl. Mech. Engrg., 192 (2003), pp. 1177–1226.
[2] H. C. Elman, V. E. Howle, J. N. Shadid, and R. S. Tuminaro, *A parallel block multi-level preconditioner for the 3D incompressible Navier-Stokes equations*, J. Comput. Phys., 187 (2003), pp. 504–523.
[3] H. C. Elman, D. J. Silvester, and A. J. Wathen, *Performance and analysis of saddle point preconditioners for the discrete steady-state Navier-Stokes equations*, Numer. Math., 90 (2002), pp. 665–688.
[4] J. Boyle, M. D. Mihajlović, and J. A. Scott, *HSL_MI20: An Efficient AMG Preconditioner*, Technical rep. RAL–TR–2007–021, STFC Rutherford Appleton Laboratory, Didcot, UK, 2007, available online at http://epubs.cclrc.ac.uk/bitstream/1961/bmsRALTR2007021.pdf.
[5] H. C. Elman, D. J. Silvester, and A. J. Wathen, *Finite Elements and Fast Iterative Solvers*, Oxford University Press, Oxford, UK, 2005.
[6] D. Kay, D. Loghin, and A. J. Wathen, *A preconditioner for the steady-state Navier–Stokes equations*, SIAM J. Sci. Comput., 24 (2002), pp. 237–256.
[7] M. F. Murphy, G. H. Golub, and A. J. Wathen, *A note on preconditioning for indefinite linear systems*, SIAM J. Sci. Comput., 21 (2000), pp. 1969–1972.
[8] P. M. Gresho, D. F. Griffiths, and D. J. Silvester, *Adaptive time-stepping for incompressible flow; part* I: *Scalar advection-diffusion*, SIAM J. Sci. Comput., 30 (2008), pp. 2018–2054.
[9] P. M. Gresho and R. L. Sani, *Incompressible Flow and the Finite Element Method: Vol.* 2: *Isothermal Laminar Flow*, John Wiley, Chichester, UK, 2000.
[10] W. Layton, *Introduction to the Numerical Analysis of Incompressible Viscous Flows*, Computational Science and Engineering 6, SIAM, Philadelphia, 2008.
[11] O. Osterby, *Five ways of reducing the Crank–Nicolson oscillations*, BIT, 43 (2003), pp. 811–822.
[12] P. N. Shankar and M. D. Deshpande, *Fluid mechanics in the driven cavity*, Annu. Rev. Fluid Mech., 32 (2000), pp. 93–136.
[13] D. J. Silvester, H. Elman, D. Kay, and A. Wathen, *Efficient preconditioning of the linearized Navier–Stokes equations*, J. Comput. Appl. Math., 128 (2001), pp. 261–279.
[14] A. Smith and D. J. Silvester, *Implicit algorithms and their linearisation for the transient Navier–Stokes equations*, IMA J. Numer. Anal., 17 (1997), pp. 527–543.
[15] J. C. Simo and F. Armero, *Unconditional stability and long-term behaviour of transient algorithms for the incompressible Navier–Stokes and Euler equations*, Comput. Methods Appl. Mech. Engrg., 111 (1994), pp. 111–154.
[16] Syamsudhuha and D. J. Silvester, *Efficient solution of the steady-state Navier–Stokes equations using a multigrid preconditioned Newton–Krylov method*, Internat. J. Numer. Methods Fluids, 43 (2003), pp. 1407–1427.
[17] S. Turek, *Efficient Solvers for Incompressible Flow Problems*, Springer-Verlag, Berlin, 1999.
[18] S. Turek, *A comparative study of some time-stepping techniques for the incompressible Navier-Stokes equations: From fully implicit nonlinear schemes to semi-implicit projection methods*, Internat. J. Numer. Methods Fluids, 22 (1996), pp. 987–1011.